Amendments to the Claims:

1. (currently amended) A computer apparatus suitable for use in a fast compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising:

a central processing unit (CPU);

a computer memory coupled to said CPU, said computer memory comprised of a computer readable medium;

a compilation program embodied on said computer readable medium, said compilation program comprising:

a first code segment that receives a class file listing including bytecode;

a second code segment that creates optimized machine code from said bytecode in a single sequential pass in which information from preceding instruction translations is used to perform the same optimizing process of an optimizing compiler without the extensive memory and time requirements ~~compiles said class file listing into machine code~~; and

a third code segment that interprets and executes said machine code.

2. (currently amended) A computer apparatus suitable for use in the first compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising:

a development or target computer system, said development or target computer system comprised of a computer readable storage medium containing a compilation program and one or more class files, said one or more class files containing one or more methods containing bytecode instruction listings;

said compilation program contained on said storage medium comprised of a first plurality of instructions, said first-plurality of instructions executed sequentially for each bytecode instruction, said first plurality configured to select first class to compile;

said compilation program contained on said storage medium comprised of a second plurality of instructions, said second plurality configured to select first method of said first class to compile;

said compilation program contained on said storage medium comprised of a third plurality of instructions, said third plurality configured to create map storage to store actual mappings and native code addresses and initialize stack mappings to ~~empty~~ "empty" and addresses to ~~unknown~~ "unknown";

said compilation program contained on said storage medium comprised of a fourth plurality of instructions, said fourth plurality configured to sequentially select each bytecode instruction in each said method of each ~~said class file~~ of said one or more class files;

said compilation program contained on said storage medium comprised of a fifth plurality of instructions, said fifth plurality configured to detect stored stack ~~maps~~ mappings for said selected bytecode instruction;

said compilation program contained on said storage medium comprised of a sixth plurality of instructions, said sixth plurality configured to detect direct control flow from a bytecode instruction previous to said selected bytecode instruction, said detection of direct control flow from said previous bytecode instruction resulting in said sixth plurality of instructions storing all stacks and setting said stack mappings to ~~stack~~ "stack", lack of said detection of said direct control flow from said previous bytecode instruction resulting in said sixth plurality of instructions reading stack layout from said stack mappings and setting said stack mappings to ~~stack~~ "stack";

said compilation program contained on said storage medium comprised of a seventh plurality of instructions, said seventh plurality configured to set said native code address for actual instruction;

said compilation program contained on said storage medium comprised of an eighth plurality of instructions, said eighth plurality configured to detect if said actual instruction is a load constant instruction, said detection of load constant instruction resulting in said eighth plurality of instructions creating new constant stack mapping;

said compilation program contained on said storage medium comprised of a ninth plurality of instructions, said ninth plurality configured to detect if said actual instruction is a load local instruction, said detection of load local instruction resulting in said ninth plurality of instructions creating new local stack mapping;

said compilation program contained on said storage medium comprised of a tenth plurality of instructions, said tenth plurality configured to detect if said actual instruction is a stack manipulating instruction, said detection of stack manipulating instruction resulting in said tenth plurality of instructions duplicating or reordering said stack mapping according to said stack manipulating instruction;

said compilation program contained on said storage medium comprised of an eleventh plurality of instructions, said eleventh plurality configured to detect if said actual instruction is a jump or switch instruction, said detection of jump or switch instruction resulting in said eleventh plurality of instructions emitting code using said stack mapping information and storing unused ~~all said~~ stack values ~~not used~~;

said compilation program contained on said storage medium comprised of a twelfth plurality of instructions, said twelfth plurality configured to detect if said actual instruction is a remaining type of instruction, said detection of remaining type of instruction resulting in said twelfth plurality of instructions emitting code using said stack mapping information;

said compilation program contained on said storage medium comprised of a thirteenth plurality of instructions, said thirteenth plurality configured to select next instruction;

said compilation program contained on said storage medium comprised of a fourteenth plurality of instructions, said fourteenth plurality configured to select next method; and

said compilation program contained on said storage medium comprised of a fifteenth plurality of instruction, said fifteenth plurality configured to select next class file.

3. (currently amended) A computer implemented method for compilation of preverified platform neutral bytecode instructions resulting in high quality native machine code, comprising ~~the steps of~~:

receiving a class file onto a computer readable medium containing compilation procedure instructions, said class file containing one or more methods containing platform neutral bytecode listing;

executing said compilation procedure instructions on said bytecode listings, said compilation procedures instructions sequentially processing each ~~byte code~~ bytecode instruction of said bytecode ~~listing~~ listings; and

producing native machine code on said computer readable medium, using preceding translation information to optimize said native machine code.

~~using information from preceding instructions to mimic an optimizing compiler; and producing native machine code on said computer readable medium.~~

4. (currently amended) A computer implemented method as recited in Claim 3 ~~wherein said compilation procedure selects~~ further comprising selecting first class to compile.

5. (currently amended) A computer implemented method as recited in Claim 4 ~~wherein said compilation procedure selects~~ further comprising selecting first method of said first class to compile.

6. (currently amended) A computer implemented method as recited in Claim ~~5 wherein said compilation procedure creates~~ 3 further comprising creating map storage to store actual mappings and native code addresses and ~~initializes~~ initializing stack mappings to ~~empty~~ "empty" and addresses to ~~unknown~~ "unknown".

7. (currently amended) A computer implemented method as recited in Claim ~~6 wherein said compilation procedure sequentially selects~~ ~~each bytecode instruction in~~ 3 further comprising sequentially selecting each said method of each said class file.

8. (currently amended) A computer implemented method as recited in Claim ~~7 wherein said compilation procedure detects stack maps for said selection~~ 3 further comprising detecting stack mappings for said sequentially processed bytecode instructions ~~instruction~~.

9. (currently amended) A computer implemented method as recited in Claim ~~8 wherein said compilation procedure detects~~ 3 further comprising detecting direct control flow from a bytecode instruction previous to ~~said~~ a selected actual bytecode instruction, said detection of direct control flow from said previous bytecode instruction resulting in storing all stacks and setting said stack mappings to ~~stack~~ "stack", lack of said detection of said direct control flow from said previous bytecode instruction resulting in reading stack layout from said stack mappings and setting said stack mappings to ~~stack~~ "stack".

10. (currently amended) A computer implemented method as recited in Claim ~~9 2herein said compilation procedure sets said~~ 3 further comprising setting a native code address for an actual instruction.

11. (currently amended) A computer implemented method as recited in Claim 10 ~~wherein said compilation procedure detects~~ further comprising detecting if said actual instruction is a load constant instruction, said detection of load constant instruction resulting in said method creating new constant stack mapping.

12. (currently amended) A computer implemented method as recited in Claim ~~11 wherein said compilation procedure detects~~ 10 further comprising detecting if said actual instruction is a

load local instruction, said detection of load local instruction resulting in said method creating new local stack mapping.

13. (currently amended) A computer implemented method as recited in Claim ~~12 herein said compilation procedure detects~~ 10 further comprising detecting if said actual instruction is a stack manipulating instruction, said detection of stack manipulating instruction resulting in said method duplicating or reordering said stack mapping according to stack manipulation instruction.

14. (currently amended) A computer implemented method as recited in Claim ~~13 herein said compilation procedure detects~~ 10 further comprising detecting if said actual instruction is a jump or switch instruction, said detection or jump or switch instruction resulting in said method emitting native machine code using said stack mapping information and storing unused ~~all said~~ stack values ~~not used~~.

15. (currently amended) A computer implemented method as recited in Claim ~~14 herein said compilation procedure detects~~ 10 further comprising detecting if said actual instruction is a remaining type of instruction, said detection of remaining type of instruction resulting in said method emitting native machine code using said stack mapping information.

16. (currently amended) A computer implemented method as recited in Claim ~~15 herein said compilation procedure selects~~ 10 further comprising selecting a next instruction.

17. (currently amended) A computer implemented method as recited in Claim ~~16 wherein said compilation procedure selects~~ 10 further comprising selecting a next method.

18. (currently amended) A computer implemented method as recited in Claim ~~17 wherein said compilation procedure selects~~ 10 further comprising selecting a next class file.

19. (currently amended) A computer implemented method as recited in Claim ~~18 wherein said compilation procedure produces~~ 3 further comprising producing said native machine code in a single sequential pass in which information from preceding instruction translations is used to perform the same optimizing process of an optimizing compiler without the extensive memory and time requirements.

20. (new) The method of claim 1, further comprising mimicking an optimizing compiler.

21. (new) A method, comprising:

processing a first bytecode of a sequence of bytecodes;

processing a second bytecode of the sequence of bytecodes using information associated with the processing of the first bytecode; and

producing optimized native machine code in a single pass through the sequence of bytecodes.

22. The method of claim 21, further comprising mimicking an optimizing compiler.

23. The method of claim 21, further comprising using preceding translation information to optimize the native machine code.

24. The method of claim 21, further comprising using information from previously processed bytecodes of the sequence of bytecodes to mimic an optimizing compiler.